实验报告

姓名：张启元
学号：2024201541

1. 数据结构设计

本实验设计了四个结构体来存储有关信息：

```
// 定义输入结构体
struct Input{
    ll pre_block;          // 该input所引用的output所在区块的高度
    string prevTxID;       // 该input所引用的output所在交易的txID
    ll prevTxOutIndex;     // 该input所引用的output位于所在交易output集合中的索引
    string scriptSig = ""; // 脚本和签名，本实验置空
};

// 定义输出结构体
struct Output{
    string txid;           // 该output所属的交易
    ll index;              // 该output在所属交易中的索引值
    ll value;              // 该output的价值(数据已乘10^8,避免浮点误差)
    string script = "";    // 脚本，本实验置空
};

// 定义交易结构体
struct Transaction{
    string txid;            // 交易编号，具有唯一性
    ll input_count;         // inputs的数量(忽略)
    ll output_count;        // outputs的数量(忽略)
    vector <Input> inputs;  // 一组input的集合，表示当前交易的输入所用到的输出
    vector <Output> outputs;// 一组output的集合，表示当前交易的输出，可能作为后续交易的输
入
    int is_coinbase;        // 表示是否为coinbase交易（1为coinbase交易，0为非coinbase
交易）
};

// 定义区块结构体
struct Block{
    ll height;                          // 当前块的高度，一条链上每个区块的Height均不
相同
    string hash;                        // 本区块的哈希值
    string prevhash = "";               // 前一个区块的哈希值，置空
    string merkleRoot = "";             // 本区块中所有交易的默克尔树根，置空
    ll nonce = 0;                       // 忽略
    vector <Transaction> transactions;  // 一组交易的集合
};
```

并使用结构体数组 vector <Block> blocks 来存储各个区块的内容。其中，Block结构体包含Transaction结构体，而Transaction包含了Input以及Output，各个Block之间虽然是存储在同一个结构体数组中，但是逻辑上通过hash变量和prevhash变量来进行“连接”，达到了链表的效果。

2. 非法交易判断

本实验设计了IsLegal函数来判断交易是否合法，主要分为以下几个步骤：

1> Coinbase交易验证

```
    if (tra.is_coinbase == 1) {
        return tra.inputs.empty();
    }
```

如果为Coinbase交易，只需考虑其有无输入即可。

2> 每个input所使用的output能否找到

```
    for (const auto &input : tra.inputs) {
        bool output_exists = false;
        for (const auto &block : blocks) {
            if (block.height == input.pre_block) {
                for (const auto &transaction : block.transactions) {
                    if (transaction.txid == input.prevTxID) {
                        // 检查引用的输出索引是否有效
                        if (input.prevTxOutIndex < transaction.outputs.size()) {
                            output_exists = true;
                        }
                        break;
                    }
                }
                break;
            }
        }
        if (!output_exists) {
            return false;  // 如果引用的输出不存在，交易不合法
        }
    }
```

对于输入交易的每个input，先找到height为其pre_block的block，在从中找到txid为prevTxID的transaction，再判断该input的prevTxOutIndex是否在该transaction的output个数范围内即可。

3> 每个input所使用的output是否被之前的交易用过

```
    for (const auto &input : tra.inputs) {
        for (const auto &block : blocks) {
            if (block.height > input.pre_block) break;
            for (const auto &transaction : block.transactions) {
                if (transaction.txid == tra.txid) continue;  // 跳过当前交易自身
                for (const auto &other_input : transaction.inputs) {
                    // 检查是否有其他交易引用了相同的输出
                    if (other_input.pre_block == input.pre_block &&
                        other_input.prevTxID == input.prevTxID &&
                        other_input.prevTxOutIndex == input.prevTxOutIndex) {
                        return false;  // 存在双花，交易不合法
                    }
                }
            }
        }
    }
```

```
    }
```

与上面的判断流程基本类似，先找到该input所引用的ouput，再遍历这之前的其它交易里的input，检查这些input是否引用了相同的output。

4> 该交易所有input所引用的output的value之和是否大于等于该交易所有output的value之和

```
    ll total_input_value = 0;
    ll total_output_value = 0;

    // 计算所有输入的总价值
    for (const auto &input : tra.inputs) {
        for (const auto &block : blocks) {
            if (block.height == input.pre_block) {
                for (const auto &transaction : block.transactions) {
                    if (transaction.txid == input.prevTxID) {
                        if (input.prevTxOutIndex < transaction.outputs.size()) {
                            total_input_value +=
transaction.outputs[input.prevTxOutIndex].value;
                        }
                        break;
                    }
                }
                break;
            }
        }
    }

    // 计算所有输出的总价值
    for (const auto &output : tra.outputs) {
        total_output_value += output.value;
    }

    // 检查输入是否足够支付输出
    return total_input_value >= total_output_value;
```

这一模块的查找方式与上面两个类似，只不过多了一步计算价值的步骤，最终比较input与output的value之和的大小即可。

3. 运行结果演示

本实验设计了显示统计信息、输出区块内容以及输出交易内容三个输出函数，在本机的运行结果如下：

Demo数据运行结果：

```
d:\vscode_cpp\Algorithm>cd "d:\vscode_cpp\Algorithm\" && g++ Lab_BlockChain.cpp -o Lab_BlockChain && "d:\vscode_cpp\Algorithm\"Lab_BlockChain
Length of Block chains: 4
Total number of Block chains: 4
Total number of legal transactions: 5
Total number of illegal transactions: 0

Please enter the block height you want to query: 2
Block Information:
Height: 2
Hash: 00000000000000000000078848a1a395bd4588381d8eb41ae8ad33dbfe9e4a38da
PrevHash: 000000000000000000000bb65617ed6bd57a9ee0afaa87fdf1e0d4061452c22442
MerkleRoot: ae290e078d5f89ef2326ba944426fdddefce46acb10a0ac6e82bd5ec9364127b
Nonce: 2223907475

Please enter the transaction ID you want to query: 5a916d9e74946ed6f3c2aec1acea20ae59a2af216eb9b33f91a0771f20678bed
Transaction Information:
TXID: 5a916d9e74946ed6f3c2aec1acea20ae59a2af216eb9b33f91a0771f20678bed
input_count: 1
output_count: 2
is_coinbase: 0
```

Data数据运行结果：

```
d:\vscode_cpp\Algorithm>cd "d:\vscode_cpp\Algorithm\" && g++ Lab_BlockChain.cpp -o Lab_BlockChain && "d:\vscode_cpp\Algorithm\"La
b_BlockChain
Length of Block chains: 32490
Total number of Block chains: 32490
Total number of legal transactions: 32705
Total number of illegal transactions: 4

Please enter the block height you want to query: 3
Block Information:
Height: 3
Hash: 0000000082b5015589a3fdf2d4baff403e6f0be035a5d9742c1cae6295464449
PrevHash: 000000006a625f06636b8bb6ac7b960a8d03705d1ace08b1a19da3fdcc99ddbd
MerkleRoot: 999e1c837c76a1b7fbb7e57baf87b309960f5ffefbf2a9b95dd890602272f644
Nonce: 1844305925

Please enter the transaction ID you want to query: 43c39b8b3728c6cb26fae0fc18803ca6cf43e15fde218e3dfbd54e633cf6753e
Transaction Information:
TXID: 43c39b8b3728c6cb26fae0fc18803ca6cf43e15fde218e3dfbd54e633cf6753e
input_count: 1
output_count: 1
is_coinbase: 1
```